TOWARD AN EMBEDDED TRAINING TOOL FOR DEEP SPACE NETWORK OPERATIONS

Randall W. Hill, Jr.
Kathryn F. Sturdevant

Jet Propulsion Laboratory
California institute of Technology
Pasadena, California

W. Lewis Johnson

University of southern California
information sciences Institute
Marina del Rey, California

Abstract

There arc three issues to consider when building an embedded training system for a task domain involving the operation of complex equipment: (1) how skill is acquired in the task domain; (2) how the training system should be designed to assist in the acquisition of the skill, and more specifically, how an intelligent tutor could aid in learning; and (3) whether it is feasible to incorporate the resulting training system into the operational environment. This paper describes how these issues have been addressed in a prototype training system that was developed for operations, in NASA's Deep Space Network (DSN). The first two issues were addressed by building an executable cognitive model of problem solving and skill acquisition of the task domain and then using the model to design an intelligent tutor. The cognitive model was developed in Soar for the DSN's Link Monitor and Control (LMC) system; it led to several insights about learning in the task domain that were used to design an intelligent tutor called REACT that implements a method called "impasse-driven tutoring." REACT is one component of the LMC training system, which also includes a communications link simulator and a graphical user interface. A pilot study of the LMC training system indicates that REACT shows promise as an effective way for helping Operators to quickly acquire expert skills. This prototype is being used to address the feasibility of embedding the training system in the DSN environment.

Introduction

The Deep Space Network (DSN) is a world-wide system for navigating, tracking, and communicating with NASA's unmanned interplanetary spacecraft. The success of the DSN in accomplishing its mission is largely a result of the constant scientific and engineering improvements that have been made to the communications devices and the systems that monitor and control them; but this success is also due to the ability of the Operators to execute complex procedures on the communications equipment in a timel y and error-free manner. Operations mistakes can be costly, both in terms of lost mission data and damage to equipment. For this reason, it is essential that the Operators be well trained and possess the skills that are required for performing tasks in a highly interactive environment. This paper addresses the issue of how to build an embedded training system that can assist Operators in rapidly acquiring the expert level of skill needed for the, DSN operations domain.

In considering how to build an embedded training system, three issues were addressed in the development of a prototype called the Link Monitor and Control (LMC) Training System: First, what is skill and how is it acquired in the task domain? The answer to this question was attained by developing a computational cognitive model of problem solving and skill acquisition in the LMC system task domain. Hill&Johnson (1993a) and Hill (1993) describe the cognitive model in detail and the results of this work arc summarized in the section entitled “A Cognitive Model of Skill Acquisition.”

Second, how should the design of the training system reflect what is known about skill acquisition in the domain? in particular, what sort of training should be provided and how can an intelligent tutor best interact with the student to optimize the learning process? Based on the results of the cognitive modeling work, a new method called “impasse-driven tutoring” was devised and implemented in a tutor called REACT, which is a component in the LMC training system prototype. A pilot study evaluating the LMC training system was conducted, and the results indicate that REACT will be an effective tutor for the LMC training system. The details of this work are reported in Hill&Johnson (1993b) and Hill (1993) and arc summarized in the section entitled "LMC Training System.”

The first two issues have largely been addressed by the research summarized in this paper, but there is a third issue that is still open, namely, how feasible is it to embed the training system in an operational environment? In many respects this will remain an open issue until an attempt has been made to actually embed the trainer in a system, but there arc a number of reasons to believe that it is at least possible, based on the design of the tutor that is described in this paper. A discussion of some of the issues that must still be addressed to prove the feasibility of building an embedded training system arc addressed both in the section on the LMC Training System and the following section on “Open Issues."

A Cognitive Model of Skill Acquisition

The ultimate purpose for building a cognitive model was to predict the effectiveness of various tutoring strategies in the LMC domain (Hill&Johnson, 1993a). In order to build the model, an analysis of the task domain was conducted; Operators were observed and interviewed and sample session logs were analyzed. The analysis provided a number of insights about the nature of knowledge and problem solving behavior at both the novice and expert levels of skill, which led to some hypotheses about skill acquisition that were implemented in an execu table cognit ive model that accounts for the transition from novice to expert, Based on the cognitive modeling work a new method tutoring st ra tegy called “impasse-driven tutoring" was developed and implemented in the REACT tutor.

Task Domain Analysis: DSN Operations

Tasks in the LMC domain involve operating a communications link in NASA’s Deep Space Network (DSN) to accomplish a mission, which is a scheduled event that typically lasts several hours. Each mission has a set of high level goals, e.g., acquire data from a deep space probe, determine the location of a spacecraft using very long baseline interferometry, etc.; and each mission also has a set of written procedures that specify actions to be taken on the communications link devices. Communications links are formed on a mission-by-mission basis; a link is typically composed of a large dish antenna (26, .34 or 70 meters in diameter), its electromechanical controllers and subsystems, a receiver/exciter, and so on. Procedures arc executed by sending commands via the LMC system to the devices assigned to the link .

Figure 1 shows an example of a typical mission hierarchy. In this case the mission is a type called Very Long Baseline Interferometry

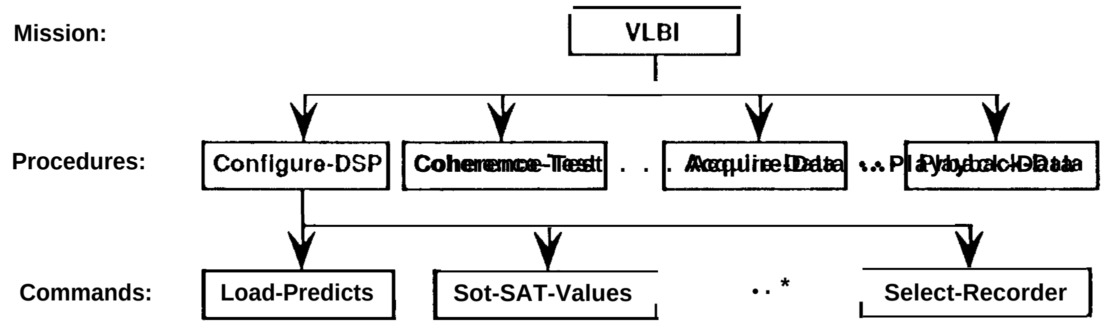


Figure 1: Example mission: Very Long Baseline Interferometry

(VLBI), which consists of a set of procedures: Configure-DSP, Coherence-Test, Acquire-Data, Playback-Data, and so on.

Each of the procedures involves issuing a sequence of commands. Figure 1 shows some of the commands for the Configure-DSP procedure. Once a command has been sent, the Operator waits for a response from the target subsystem that indicates whether the command was rejected or accepted. If the command is accepted, the Operator monitors the subsystem to determine also whether the command had the desired effect.

Here is a summary from (Hill&Johnson, 1993a; Hill, 1993) of what was learned about problem solving and skill in the LMC task domain:

1. *Rote procedure execution* is *not sufficient*. Through interviews with LMC Operators and anal ysis of LMC Operator logs, it became clear that procedures cannot always be executed in the order specified. The devices being controlled are interactive in nature, consequently, changes in state are not instantaneous when a command is issued. Furthermore, devices may not be in an optimal state for taking an action specified by the procedure, and devices may fail unexpectedly, leading to the need to change the default procedure. This leads to the next two observations.

2. *Commands have preconditions that must be satisfied before they can be successfully executed.* For this reason, a command cannot be blindly executed, rather, the Operator must verify that the associated devices are in the right state before issuing it. Thus a precondition is a description of a device state, and each command can have multiple preconditions that must be satisfied. When the Operator executes a command without regard to its preconditions, the command may either be rejected or lead to a system failure or error. At a minimum, a command rejection costs the Operator time; some errors may lead to the failure of a procedure's goals.

3. *Commands have effects, or postconditions, whose successful implementation must be verified once they have been issued.* In other words, the Operator must attend to the effects of a command once it has been issued. Like a precondition, a postcondition is also a description of a device state; it can be viewed as a goal that must be satisfied in order to complete the intention of the command. So not only does the Operator have to attend to the state of the devices prior to issuing a command, but the stale of the devices must also be attended to afterward as well.

4. *Mission procedures have goals.* Each of the procedures that are used in the performance of a mission has a set of goals that must be accomplished. If a procedure has been completely executed and its goals have not been achieved, then the Operator must take remedial actions to accomplish them. Of course, if the Operator does not recognize that the procedure's goals have not been achieved then it is possible that the mission's goals may not be achieved. For example, if a parameter setting on a device is given the wrong value by the Operator then the data that is collected may be affected, making it more difficult for the scientist to process and use it.

5. *A partial order exists among procedures.* This is an artifact of the DSN operational environment where a strict order is not always defined among procedures (Fayyad&Cooper, 1992; Hill&Lee, 1992). It is possible and often desirable to interleave procedures in order to optimize the execution of the mission task, but there are dependencies among the procedures that compel the Operators to partially order their execution. There are two problems that arise when the Operator does not understand the dependencies among procedures: either the task is performed inefficiently, i.e., by not interleaving the procedures, or else the procedures are performed out of order, which may lead to errors and lost time.

A Theory of Skill Acquisition

An executable cognitive model was developed to reflect the observations that are outlined above. By an executable cognitive model, it is meant that the model produces behavior similar to what has been observed in actual DSN operations. Since the purpose of the cognitive model was to inform the design of an intelligent tutor, there were two additional goals for its implementation. First, it had to reflect the behaviors of both novice and expert problem solvers. Second, it had to lead to hypotheses about how a novice acquires skill in route to becoming an expert.

The cognitive model was developed in Soar, which is a problem solving architecture based on a theory of cognition (Laird et al., 1987; Newell, 1990). Soar has a learning mechanism called "chunking" that generates new productions to summarize the results of problem solving in a goal hierarchy (Rosenbloom&Newell, 1986). Subgoals arc formed when Soar reaches an impasse, of which there arc several types. A Soar impasse occurs when the problem solver is not making progress toward a solution. If a solution is found by the subgoal, then a chunk is created to summarize the conditions that led to the impasse and the results that resolve it, so that the next time the impasse conditions occur the chunk can be applied.

A part of the cognitive model is shown in Figure 2. in this example the procedure called Configure-IXI' is modeled, which is onc of the procedures from the VLBI mission that is shown in Figure 1. This procedure involves performing actions to load a predicts file (Load-Predicts), select a recording device (Select-l{ccordcr), and so on. Thus the boxes that are shown next to the "Commands" label in Figure 2 are all considered to bc operators, that is, functions that have preconditions and postconditions. The example in Figure 2 shows the cognitive mode] issuing the command associated with setting the S-Band attenuation value, SAT 30, which is done once the preconditions for the Set-SAT-value operator are satisfied.

'l'asks are performed by the Soar-based cognitive model by first selecting a procedure (e.g., Configure-DSI') and then selecting an operator to apply such as Set-SAT-Value. Each of these selections corresponds to a subgoal in the Soar architecture. Once a command operator has been selected, a subgoal is formed to verify the operator's preconditions (i.e., the subgoal called Verify-Opcratc) r-I'reconditions shown in Figure 2). If all of the preconditions arc satisfied then the model issues the command (e.g., SAT 30) to the device. Under normal circumstances the model would then have a subgoal to verify that the command had executed as expected (Verify-Opera tor-Postconditions), and would move onto the next command operator once the postconditions are satisfied. This summarizes how problem solving takes place under ideal conditions, Now consider how the model represents the behavior of an LMC Operator when the conditions are not ideal.

When an expert Operator notices that a precondition of a command operator is not satisfied then a subgoal called Repair-Unsatisfi ed-I'recondition is formed to find a command whose execution will satisfy the precondition by putting a device into a desired state, Once located, the repair operator is subjected to the same kind of precondition and postcondition checking that has already been described. It is executed and then the previously unsatisfied precondition of the original command is reevaluated. This simple form of recovery represents the reactive nature of problem solving that takes place in the LMC domain, and it illustrates why rote execution is not sufficient since the procedures do not contain all of the contingencies that may occur during a mission. This also illustrates the usc of command preconditions and postconditions by the Operators in performing a task.

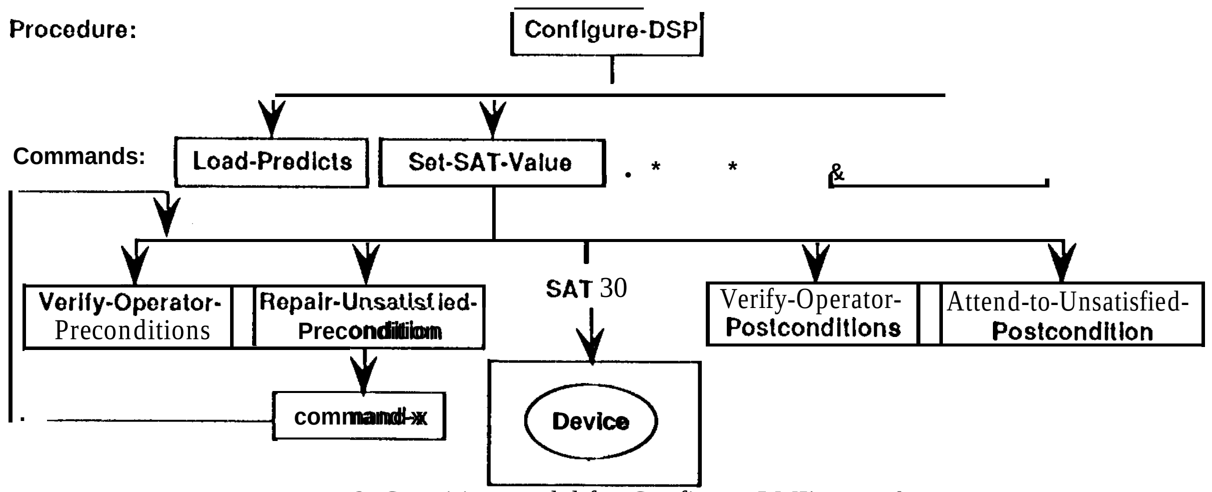

Procedure:
Configure-DSP
Commands:
Load-Predicts
Set-SAT-Value
Verify-Operator-Preconditions
Repair-Unsatisfied-Precondition
SAT 30
Verify-Operator-Postconditions
Attend-to-Unsatisfied-Postcondition
command-x
Device

Figure 2: Cognitive model for Configure-LMI' procedure

in terms of skill acquisition, the model that has been discussed thus far takes advantage of the Soar chunking mechanism. Since the problem solving is done in a goal hierarchy, chunks are being built as the problem solving proceeds. This means that the processes of selecting command operators, verifying their preconditions, repairing unsatisfied preconditions, issuing the command, and verifying its postconditions are all compiled into chunks that enable the procedure to be performed more efficiently the next time the task is attempted, This accounts for the kind of efficiency gains that arc observed among Operators who practice the task. What is still left to show is how novices are modeled, both in terms of their behavior and how they acquire new knowledge.

Novices were modeled as problem solvers who lack knowledge about command preconditions. Consequently, if the novice does not know a command's precondition and the precond it ion is not satisfied, then the novice will issue the command without attempting to repair the unsat isfied precondi t ion in the manner that the expert would. If a precondition is not satisfied the command will likel y be rejected by the system; this is the point at which the novice can potentially acquire some new knowledge. The model assumes that the Operator will notice that the command has been rejected and will attempt to do something about it; this is represented by the subgoal called Attend-to-Unsatisfied-1'ostcondition in Figure 2. In this subgoal the novice acquires knowledge about the missing precondition that led to the command reject ion. Once acquired, the precondi tion can be verified and repaired, which leads to correct behavior. As was the case with all of the problem solving, this new knowledge is compiled into chunks that improve the overall efficiency of task execution.

Design Requirements Derived from Model

The cognitive model led to a number of insights that affected the design and implementation of the LMC Training System, and especially the REACT intelligent tutor, Some of the design decisions that were made from the lessons learned from the cognitive model arc summarized in the following paragraphs, (For more details, see Hill&Johnson, 1993a, Hill&Johnson, 1993b, and Hill, 1993.)

1. *Perform the training on a simulator.* This addresses several of the results of the cognitive model. First, the cognitive model improved its performance by compiling its problem solving experience into new chunks. This is "learning by doing," and it implies that practice is required in order to acquire this aspect of skill. Second, practicing the procedures cm a simulator reproduces the interactive nature of problem solving in the LMC task domain. The cognitive model gained "reactive" skill by recognizing and repairing unsatisfied preconditions.

2. *Interact with the student when there* is a *problem solving impasse.* The cognitive model acquires new knowledge when it has a "knowledge-level" impasse (Hill, 1993). This is an impasse where problem solving cannot be continued duc to a lack of some knowledge. In the cognitive model the missing knowledge is command preconditions, but this can be expanded to include knowledge about procedural dependencies and procedure goals also. There are two reasons for interacting with the student al an impasse point. The first reason is based cm the way Soar works: learning (is., chunking) occurs in a goal context, Thus, it makes sense that tutoring will be most effective if it is applied in the goal context where the knowledge is needed, This leads to the second reason: the cognitive model acquires new knowledge at the impasse point. This was a natural place for the problem solver to seek after new knowledge since the problem solving could not continue without it.

3. Develop *a flexible recognition strategy.* This design requirement is derived from the previous two and it refers to the task of modeling the student by interpreting the student's actions. Since tasks are interactive, this means that the tutor must be capable of recognizing not only when the student has reached an impasse, but also when the student has correctly taken an action that deviates from the default procedure. It is not desirable to interrupt the student when there is not an impasse, since it can distract from the current goal context. Since the situation may dictate that a precondition be satisfied (i.e., repaired) before continuing with the default procedure, the tutor must be able to distinguish this type of behavior from impasse-based behaviors.

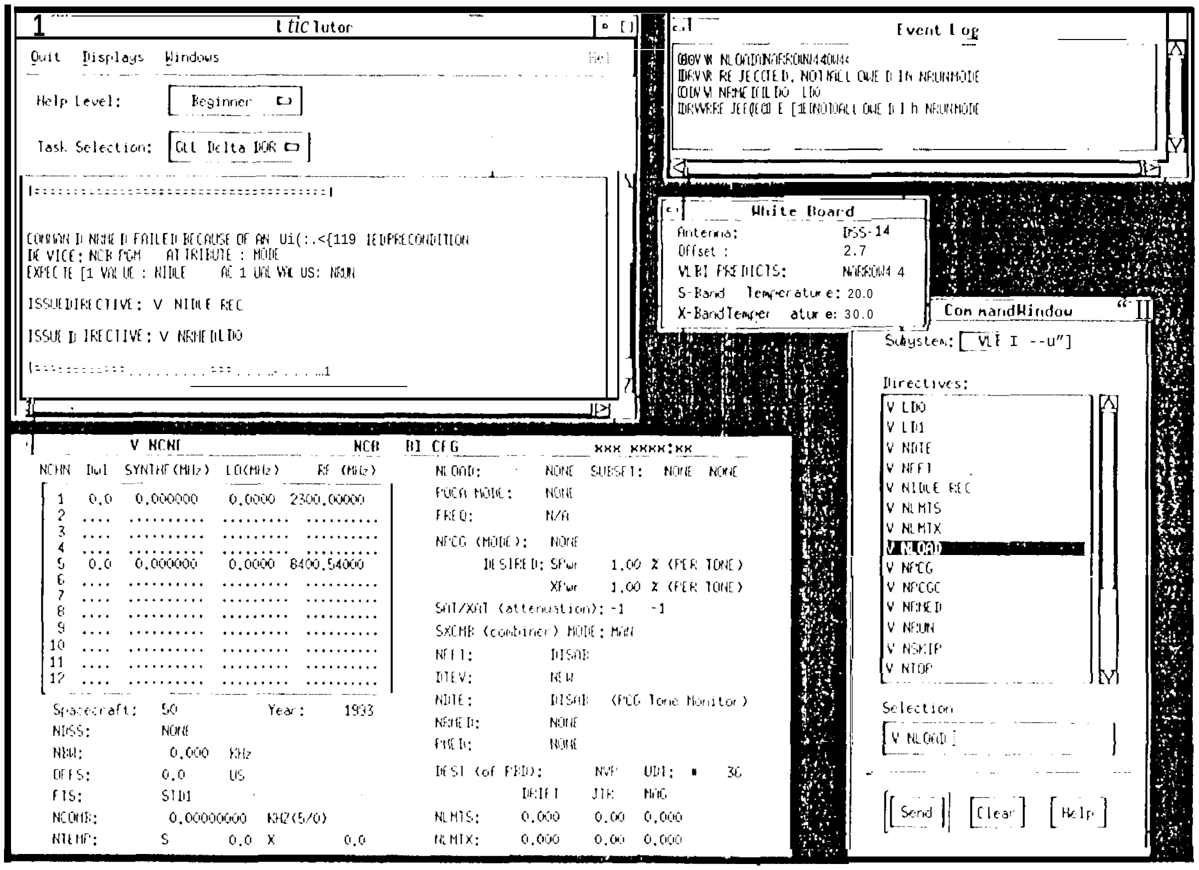
4. *Assist the student in acquiring knowledge about command preconditions and postcondi-*

tions, procedural dependencies, and procedural goals. This partly follows from the cognitive model, which shows how Operators acquire knowledge about command preconditions. Missing knowledge in the other categories will lead to impasses other than the ones that were modeled. This puts a further requirement on the student modeler to recognize when the student has an impasse due to a lack of knowledge about each of these types of knowledge.

LMC Training System

This section addresses the second issue of implementing an embedded training system, namely, how to design and implement the system to meet the requirements derived from the cognitive model. Four general requirements derived from the cognitive model were described in the last section. These requirements have driven the development of the LMC Training System and REACT intelligent tutor that will now be described. (See Figure 3.)

The requirement for a simulator is satisfied by two of the system's components, the Graphical User Interface, and the Link simulator. These components simulate the LMC user interface and the link devices' states, respectively. The other requirements are addressed by an intelligent tutor called REACT. REACT implements a method called "impasse-driven" tutoring, whereby interactions with the student are initiated when it recognizes a problem solving impasse. Impasse recognition is performed using a new technique called "situated plan attribution," which provides a flexible way of tracing the student's behavior: it can recognize when the student is deviating from a procedure in reaction to a situational contingency (Hill&Johnson, 1993b; Hill, 1993). in addition, it recognizes impasses stemming from knowledge-level gaps concerning command pre-conditions, procedural dependencies and procedural goals.



```
Help Level:      Beginner
Task Selection:  GLL Delta DOR

COMMAND NAMED FAILED BECAUSE OF AN Ui(:.<{119 IEDPRECONDITION
DEVICE: NCB PGM     ATTRIBUTE : MODE
EXPECTED VALUE : NIDLE      ACTUAL VALUE: NRUN

ISSUE DIRECTIVE: V  NIDLE REC

ISSUE DIRECTIVE: V  NAMEDLDO
```

```
Event Log
CONV NLOADNARROW440W44
DRVR REJECTED, NOT ALLOWED IN NRUNMODE
CONV NAMEDLDO  LDO
DRVREJECTED, NOT ALLOWED IN NRUNMODE
```

```
White Board
Antenna:                  DSS-14
Offset :                  2.7
VLBI PREDICTS:            NARROW4 4
S-Band  Temperature:      20.0
X-Band Temperature:       30.0
```

```
CommandWindow
Subsystem: [VLBI --u"]
Directives:
V LDO
V LD1
V NDTE
V NFFT
V NIDLE REC
V NLMTS
V NLMTX
V NLOAD
V NPCG
V NPCGC
V NAMED
V NRUN
V NSKIP
V NTOP
Selection
V NLOAD
```

```
V NCNF                                    NCB  BI CFG               XXX XXXX:XX
NCHN  Dwl  SYNTHF(MHz)  LO(MHz)    RF (MHz)      NLOAD:          NONE  SUBSET:  NONE  NONE
  1   0.0  0.000000     0.0000   2300.00000      PCCA MODE:      NONE
  2   .... ..........   ........ ..........      FREQ:           N/A
  3   .... ..........   ........ ..........
  4   .... ..........   ........ ..........      NPCG (MODE):    NONE
  5   0.0  0.000000     0.0000   8400.54000            DESIRED: SPwr    1.00 % (PER TONE)
  6   .... ..........   ........ ..........                     XPwr    1.00 % (PER TONE)
  7   .... ..........   ........ ..........      SAT/XAT (attenuation): -1    -1
  8   .... ..........   ........ ..........      SXCMB (combiner) MODE: MAN
  9   .... ..........   ........ ..........      NFFT:           DISAB
 10   .... ..........   ........ ..........      DTEV:           NEW
 11   .... ..........   ........ ..........      NDTE:           DISAB   (PCG Tone Monitor)
 12   .... ..........   ........ ..........      NAMED:          NONE
 Spacecraft:    50           Year:     1993      PMED:           NONE
 NDSS:          NONE
 NBW:           0.000    KHz                     DEST (of PRD):      NVP    UDT:  *    36
 OFFS:          0.0      US                                    DRIFT    JIT    MAG
 FTS:           STD1                             NLMTS:        0.000    0.00   0.000
 NCOMB:         0.00000000   KHZ(5/0)            NLMTX:        0.000    0.00   0.000
 NTEMP:         S          0.0   X        0.0
```

Figure 3: LMC Training System

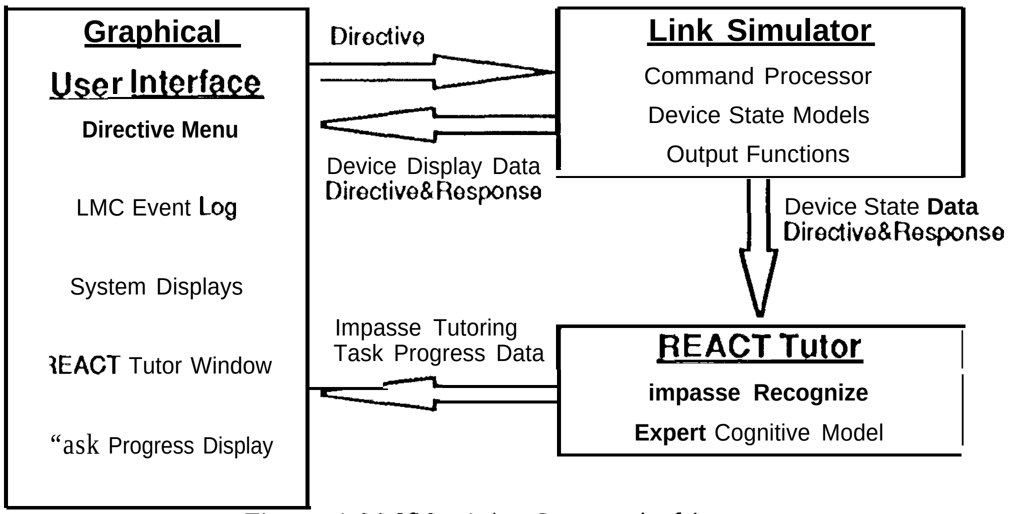


Figure 4: LMC Training System Architecture

The remainder of this section describes the LMC Training System, beginning with an example of a training session with a student. This is followed by a high-level overview of the architecture and a description of each of the training system components. The final "embedded training" issue is addressed indirectly here in that the design of the prototype system has a bearing on the later discussion on the feasibility of incorporating the trainer into an operational environment.

Example of Training Session

To better understand how the training system works, consider the example shown in Figure 3. This figure shows part of the actual graphical user interface for the training system. The Operator selects a command from the Command Window and sends it to the simulator. The simulator processes the command and sends the directive and its response to the Event Log window. Device state changes arc also sent to a system display, which in this case is the NCNF display.

in the example shown, the Operator has unsuccessfull y sent the both the N LOAD and NRMED commands, which were rejected. REACT observed both of these rejections and recognized that the Operator was at an action constraint impasse in each case. The impasses were explicated using the expert cognitive model; REACT generated explanations of what caused the commands to be rejected and how to fix the underlying problem. These explanations were sent to the LMC Tutor window. Figure 3 shows the advice that was given to the student about the impasse related to the NRMED command.

Architecture

The LMC Training System architecture, shown in Figure 4, has three components: the Graphical User Interface, the Link simulator, and the REACT tutor. The Operator interacts with the Link simulator via the windows provided by the Graphical User Interface (GUI). The simulator, in turn, communicates the device state data and directives and their responses to the REACT Tutor and back to the Graphical User Interface. Based on the Operator's actions and the device state changes, REACT determines whether the student is at an impasse. As was shown in the example, the tutor communicates its advice to the student through the Graphical User Interface. in addition, the tutor also updates a task progress display indicating how well the student is performing the task.

The LMC training system has an object-oriented design. Designing the system in this manner allows parts of the system to be reused. For example, the graphical user interface is a separate object so program code that implemented it is localized and not interwoven with other parts of the system. If it was decided to change the user interface, the GUI object could be up dated or replaced while the interfaces to the other objects would remain unchanged. Another example would be to replace the simulator object with an interface to the real LMC, while the other LMC trainer objects remained the same. Even if the LMC trainer remained separate from the LMC system, the user interface object could be shared between the trainer and the LMC, and therefore Operators would be receiving training cm the user interface with which they would eventually operate the system.

Link Simulator
The Link simulator models the state of the devices in the communications link. It is designed to model these devices from an Operator perspective: it reacts to Operator commands with the same observable state changes and responses as the actual devices. The three functional components of the simulator are the device models, a command processor, and a set of output functions. The device models keep track of the current states of the devices. The command processor parses the Opera tor commands and causes the devices to change state. The output functions generate messages to the operator that appear in the Event Log and System Displays.

Graphical User Interface
The graphical user interface (GUI) for the LMC training system was developed using the Motif^{TM} widget set and the Widget Creation Library (WCL).* The Motif widget set was chosen because it has become a standard for user interface development. WCL was chosen to make the user interface development easier and faster. Because WCL allows the user to define the entire widget tree in resource files, it reduces the amount of code and thus time required to develop the user interface.

The GUI was designed to be easy to use: the operator inputs commands to the system from a menu, where the commands are provided in an alphabetical list. The user select and clicks on a command with a mouse pointing device and types in any required parameters. (See the Command Window in Figure 3.) As with the real LMC, the event log is very important to the operator. The LMC trainer has a representation of the event log, which prints the operator directives and the system responses. (See the Event Log window in Figure 3.) When the LMC trainer detects an impasse, it prints the problem and suggested solution in the LMC Tutor window, shown in Figure 3. The LMC Tutor window also provides access to other LMC displays through pull down menus. An example of an LMC display is the NCB Configuration Table (NCNF) shown also in Figure 3. Other information that operators need is written on white boards in the operations room. This information is represented in the LMC trainer as the White Board display. There arc many more LMC displays available to the operator which are not yet represented in the trainer. Figure 3 represents a sample layout of the LMC tutor windows. The operator can arrange the windows to his/her liking.

REACT Tutor
The tutor monitors the state of the devices in the Link simulator and all student-sirnulator command interactions. The student performs an assigned task and REACT uses a technique called “situated plan attribution” to interpret the student’s behavior (Hill&Johnson, 1993b; Hill, 1993).

Since the tutoring is impasse-driven, it is imperative to recognize when the student has reached an impasse. REACT recognizes impasses in three categories: action-constraint, plan dependency, and goal failure, The theory behind this method is that the impasses in each of the.sc categories occur as a result of a know]edge-level gap or misconception about command preconditions/postcon ditions, plan dependencies, and plan goals, respectively. (Note: the terms “plans” and “procedures” are used synonymously.) Once an impasse is detected, the tutor employs its expert cognitive model to generate an explanation of the impasse and how to resolve it. This information is given to the student through the LMC Tutor window.

A pilot study has been conducted to determine the effectiveness of the REACT tutor in helping Operators acquire skill in the task domain (Hill, 1993). This evaluation indicates that students who usc the LMC training system with the tutor arc able to resolve impasses during training approximately ten times as fast as students who use it without the tutor. In addi-

* The Widget Creation Library (WCL) was developed by David E. Smyth at the Jet Propulsion Laboratory. It is available by anonymous ftp from export.lcs. init. edu in the files contrib/Wcl-2.5 .tar. Z with documentation in the files contrib/WclDoc.t.Z and contrib/Wcl7XTC.ps.t.Z.

tion, the tutor detected impasses that went unnoticed by the untutored students; the untutored student could fail to achieve a goal and never realize it, which is one of the problems that currently exists in the DSN operational environment.

Open Issues

There arc many ways of envisioning how a training system such as the one described in this paper could be integrated into an operational environment, but for the purposes of this discussicm only two cases arc considered: (1) embed the training system in the operational system (e.g., in the actual LMC), or (2) keep the training system as a separate entity.

in case 1, where the training system is an integrated part of the operational system, the question that needs to be addressed first is whether it is practical or desirable to conduct training on the actual system. If there is no time available for using the operational system for training, then it probably does not make sense to embed a training system in it. On the other hand, if a flexible view of training is taken where the Operator uses "dead time" during a track to practice for an upcoming mission then it may prove useful to provide this capability.

In case 2, where the training system is a separate entity, there are other operational issues: 1s the system located in a place where it will be used? Will the operators be allocated the time to use the training system? These questions arc beyond the scope of this research, but they have a bearing on deciding whether to implement the trainer as an embedded part of the operational system or as a separate entity.

Given that case 1 or case 2 makes operational sense, then the feasibility of implementing the trainer for the operational system can be evaluated by asking the following questions with respect to each of the trainer's components: (1) 1s the component needed? (2) How will the component be built so that it accurately reflects the target system? (3) How will the component be maintained to reflect changes to the target system?

Clearly, all three of the training system prototype's components are conceptually necessary: the graphical user interface and simulator arc crucial for providing a realistic training environment, and the tutor provides a way of accelerating the learning process. Thus the answer to question 1 appears to be "yes" for all of the components.

The answers to questions 2 and 3 are not as clear as for question 1. These would not be issues for the graphical user interface if the training system was able to use the same code as the actual system. The trainer and the operational system would always be consistent with one another if this were the case. The only difference would be that there would be some additional windows through which the tutor could communicate with the student.

There is more of a problem with building and maintaining a simulator that is functionally consistent with the actual devices and subsystems. Clearly there would be a need to have a rigorous way of specifying the functional design so that it could be readily implemented and verified in both the simulator and the actual system, One approach might be to use the simulator as a living functional specification. Not only would this serve the needs of the training system, but it would also be helpful for testing the interfaces among subsystems. This is an area that requires further research.

The REACT tutor itself would have to be designed and maintained to accurately reflect the preconditions and postconditions of each of the commands, the dependencies among procedures, and the goals of each of the procedures. Each time the characteristics of a device or subsystem are modified, then it would be necessary to rc-evaluate the validity of the tutor's knowledge base. Again, this is an area that is open to further investigation, Because of the way that the tutor is implemented, the knowledge bases containing this kind of information are separable from the underlying method for detecting and explicating impasses. This lends itself to constructing knowledge base libraries that could be globally checked and maintained by system developers.

Finally, there is a broader issue than the two cases concerning whether the training system should be embedded or kept separate; this is the issue of utility: Does the training system help the operators to an extent that it justifies the effort needed to build it? This can be answered in part by evaluating the training system's effectiveness in training Operators. The pilot study mentioned previously provided some interesting results that appear to indicate that the training system holds promise as a means of accelerating the skill acquisition process, but the evaluat ion needs to be expanded to include a larger test population cm a larger set of tasks. ~'here are plans to perform a more rigorous evaluation of the system.

References

(Fayyad&Cooper, 1992) Kristina Fayyad and Lynne Cooper. Representing operat ions procedures using temporal dependency networks. *Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations,* SPACEOPS-92, Pasadena, CA, November 16-20,1992.

(1 lill, 1993) Randall W. Hill, Jr.. "Impasse-driven tutoring for reactive skill acquisition." Ph.D. diss. University of Southern California, Los Angeles, California, 1993.

(Hill&Lee, 1992) Randall W. Hill, Jr. and Lorrine Lee. Situation management in the Link Monitor and Control Operator Assistant. *Proceedings of the Second International Symposium on Ground Data Systems for Space Mission Operations,* SPACEOPS-92, Pasadena, CA, November 1620,1992.

(1 lill&Johnson, 1993a) Randall W. Hill, Jr. and W. Lewis Johnson. Designing an intelligent **tutoring system based on a reactive model of skill acquisition.** *Proceedings of the World Conference on Artificial Intelligence in Education (AI-ED 93),* Edinburgh, Scotland, 1993!

(Hill&Johnson, 1993b) Randall W. **Hill**, Jr. and W. Lewis Johnson. lrnpassc-driven tutoring for reactive skill acquisition. *Proceedings of the 1 993 Conference on Intelligent Computer-Aided Training and Virf ual Environmen f Technology (ICAT-VET-93),* NASA/Johnson Space Center, Houston, Texas, May 5-7, 1993.

(1 .aird et al., 1987) John E. Laird, Allen Newell and Paul S. Rosenbloom. Soar: An architecture for genera] intelligence. *Artificial Intelligence, 33,* 1987:1-64.

(Newell, 1990) Allen Newell. ***Unified Theories of Cognition.*** Harvard University Press, 1990.

(Rosenbloom&Newell, 1986) Paul S. Rosenbloom and Allen Newell. The chunking of goal hierarchies: A generalized model of practice. *Machine Laming, Volume II.* Edited by Ryszard S. Michalski, Jaime G. Carbonell, and Tom M. Mitchell. Los Altos, CA: Morgan Kaufmann Publishers, Inc., 1986:247-288.

Acknowledgements

The research described in this paper was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

Dr. Johnson was supported in part by the Ad vanccd Research Projects Agency and the Naval Research Laboratory under contract number N00014-92-K-2015 (via a subcontract from the University of Michigan). Views and conclusions contained in this paper are the authors' and should not be interpreted as representing the official opinion *or* policy of the U.S. Government or any agency thereof.